REMARKS/ARGUMENTS

These remarks are made in response to the Office Action of November 28, 2005 (Office Action). As this response is timely filed within the three-month shortened statutory period, no fee is believed due.

At page 2 of the Office Action, each of the claims were rejected on new grounds distinct from those previously asserted. Claims 1-6, 10, 11, 13-20, and 24-33 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent Number 5,920,725 to Ma, *et al.* (hereinafter "Ma") in view of U.S. Patent No. 6,314,088 to Yamano (hereinafter "Yamano"). Claims 7 and 21 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Ma in view of Yamano and in further view of Andrew S. Tanenbaum, "Computer Networks," 1996, Prentice Hall PTR, third ed. (hereinafter "Tanenbaum"). Claims 8, 9, 22, and 23 were rejected under 35 U.S.C. §103(a) as being unpatentable over Ma in view of Yamano and further in view of Applicants' Admitted Prior Art.

Independent Claims 1, 15, 24, 28, and 32 have been amended to further emphasize certain aspects of Applicants' invention. Dependent Claims 2, 16, and 33 also have been amended, while Dependent Claims 25 and 29 have been cancelled, so as to maintain consistency among the claims. As discussed herein, the amendments are supported throughout the Specification. (See, e.g., Specification, p. 13, line 19 – p. 15, line 5.) No new matter has been added by virtue of the claim amendments presented herein.

Applicants' Invention

It may be helpful at this juncture to reiterate certain aspects of Applicants' invention prior to addressing the cited references. One embodiment of the invention, typified by independent Claim 1, as amended, is a method for distributing real-time updates to active application components in an active client position. The method can include establishing a first communications connection between a platform managing the

active application components and a configuration client, where the platform and configuration client are both disposed in a client position. The method also can include establishing a second communications connection between the configuration client and configuration server, with the configuration client submitting one or more queries to the configuration server via the second communications connection.

Additionally, the method can include delivering updates to the configuration client over the second communications connection in response to at least one such query, wherein each update corresponds to at least one particular application component, and notifying the platform that updates are available. In response to the notification, execution of the particular active application components can be terminated. The terminating step according to the method, more particularly, can comprise terminating instances of each identified application component before the instances self-terminate. (See, e.g., Specification, p. 14, lines 10-19.) According to the method, the terminating step further can include removing interdependencies between the terminated application component instances and other application components. (See, e.g., Specification, p. 14, line 20 - p. 15, line 5.) The method additionally can include delivering each update over the first communications connection to the platform, applying each update to at least one corresponding application component, and re-executing each updated application component.

The Claims Define Over The Prior Art

As already noted, independent Claims 1, 10, 15, 24, 28, and 32 were each deemed unpatentable over Ma in view of Yamano. Applicants respectfully submit, however, that there are important distinctions between Applicants' invention and both Ma and Yamano.

Ma is directed to modifying a distributed client-server application as the application is executing. (Col. 4, lines 36-41; Abstract.) An application, according to Ma, is modified when a user requests the creation or modification of object class definitions in

the application stored on a server (i.e., "server machine"). A "meta server," running on the server, modifies source code in response to the user request in Ma. (Col. 6, lines 31-39.) Modification of an object class with Ma entails an object adaptor receiving a list of the modified object classes and sending a notification to an object cache. (Col. 8, lines 11-19.) Change notifications in Ma are sent from the server machine by the object adaptor, residing on the server, to object caches both on the server machine and on its clients. (Col. 8, lines 20-24; see also FIG. 5.)

Yamano is directed to a node configuration setup system comprising a configuration client node connected to a connection-oriented network and a number of configuration server nodes interconnected via the network. (Col. 3, lines 42-55; see also FIG. 1.) The setup system of Yamano operates by having servers locate client data by searching throughout the connection-oriented network. (See, e.g., Col. 4, line 30 – Col. 6, line 25; Col. 7, line 29 – Col. 9, line 7 ; see also FIGS. 3, 4, 6-8, and 10-12.)

At pages 3-4 of the Office Action, Yamano is cited as teaching a configuration client disposed in a client position, the configuration client submitting at least one query to the configuration server. Ma is cited as disclosing each of the other features recited in independent Claims 1, 10, 15, 24, 28, and 32.

Applicants respectfully maintain, however, that neither Ma nor Yamano discloses every feature recited in independent Claims 1, 10, 15, 24, 28, and 32. For example, neither reference, either alone or in combination, teaches or suggests a process of updating application components that entails terminating the execution of active application components, as recited in each of the independent claims.

At page 3 of the Office Action, it is asserted that Ma discloses the step of terminating the execution of a particular active application component as part of a procedure for updating an active application component. Applicants respectfully submit, however, that the explicit language of the reference reveals that Ma's procedure, in fact, does not involve or suggest the termination of an active application component. Contrary

to the feature recited in each of the independent claims, an active application in Ma is not terminated before the self-termination of the component. Instead, with Ma, an active application continues to operate until it reaches a predetermined point of self-termination. This difference between Applicants' invention and Ma is made explicit in the portion of the reference cited at page 3 of the Office Action, where Ma's update procedure is expressly described by the following language:

> "Object adaptor 80 is used as a proxy by the meta server to notify objects when updates are needed. Object caches subscribe to object adaptor 80 for a particular class by executing the function:
>
> objClass.addObjectListener(CacheAdaptor);
>
> This adds the client object cache as a listener for the class. When meta server 70 updates an object class, object adaptor 80 sends a notification to the object caches when the meta server calls the cache adaptor function:
>
> CacheAdaptor.objectChanged(objClass);
>
> The object caches then invalidate the entry for the object class, and for all objects in that class, by clearing a valid bit in the cache. Any new object instances are generated from the new class definition, reloading the cache with the updated object class definition in the process. Old instances of the invalidated objects remain in use until their reference counts go to zero, when they are deleted. Another client object which is referencing an invalid object can read the invalid bit from the cache and decide to release the invalid object and load the updated object.
>
> Another object that is referencing the object has the responsibility of determining when the object has been invalidated. Objects can read the valid bit or those of other objects by executing the function:
>
> obj.isInvalid();
>
> which reads the invalid bit in the object (obj) itself. An invalid object does not have to be refreshed immediately, although that is the preferred behavior. The object can continue operating, even though it has not yet

> been updated. For the example of adding the cell-phone field to a database, old objects can continue to access the old fields of the database. The cell-phone field is simply blank or invisible to the old object. However, no persistence operations, such as saving an attribute, state, or object, are allowed. An exception is raised if any persistence operation is performed by an invalid object." (Col. 9, lines 6-43.) (emphasis supplied.)

If, as implied in the Office Action, Ma's changing of an object class is assumed to be equivalent to the updating of active application component, it nevertheless is clear from the quoted language that Ma does not terminate the execution of an active component in effecting the update. In particular, it is logically impossible for the old, pre-update object in Ma to be terminated because, as explicitly described by Ma, the old object "can continue operating." The object does not terminate before self-termination, as recited in independent Claims 1, 10, 15, 24, 28, and 32. To the contrary, each of the old objects "continues operating" up until the point that their "reference count go to zero;" that is, until each self-terminates. The is the opposite of the explicit feature recited in independent Claims 1, 10, 15, 24, 28, and 32.

The distinction between Applicants' invention and Ma is made even starker by the fact that Ma allocates to other objects that reference the old object "the responsibility to determine when [an] object has been invalidated" by an update to the object. Were the old object, treated here as the equivalent of the active application component, to be terminated when effecting the update, there would be absolutely no need to allocate this responsibility because the object would have been effectively terminated. As it is, with Ma, updating relies on the clearing of a valid bit in the cache. Another object that references the old object reads the invalid bit in the object itself and thus ascertains that an update has occurred. As already noted, however, the old object continues to operate until self-terminating when its reference count goes to zero. Again, this is the opposite of terminating an active application component prior to the component's self-termination, as recited in each of the independent claims.

Moreover, the respective functions of Ma's object adaptor and Applicants' platform are entirely distinct. The platform recited in independent Claims 1, 10, and 15 manages active application components. By contrast, Ma's object adaptor 80 establishes a connection between a server and client, not two distinct client entities. Ma's object adaptor does not mange entities on the client side, as with Applicants' platform. Rather, Ma's object adaptor sends notices of server-side modifications to object classes, the object adaptor sending the notices to the clients from the server. (Col. 8, lines 20-24.)

Yamano likewise fails to teach or suggest a platform that manages active application components on a configuration client. The configuration client node in Yamano merely stores configuration data and examines data content so as to perform the functionality of a router. (See, e.g., Col. 3, lines 51-55 and Col. 3, line 64 – Col. 4, line 3.) Accordingly, neither Ma's object adaptor nor Yamano's configuration client node are comparable to Applicants' platform in terms of their functions. Accordingly, both Ma and Yamano each fail to teach or suggest a platform for managing active application components, as recited in Claims 1, 10, and 15.

Applicants respectfully submit that, whereas both Ma and Yamano each fails to teach or suggest every feature recited in independent Claims 1, 10, 15, 24, 28, and 32, the prior art fails to anticipate the independent claims. Applicants further respectfully assert that since each of the remaining dependent claims depends from one of independent Claims 1, 10, 15, 24, 28, and 23, while reciting additional features, the prior art likewise fails to anticipate any of the remaining claims.

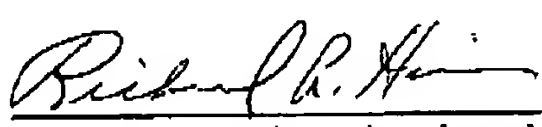
CONCLUSION

Applicants believe that this application is now in full condition for allowance, which action is respectfully requested. Applicants request that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the

Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Respectfully submitted,

Date: February 28, 2006



Gregory A. Nelson, Registration No. 30,577
Richard A. Hinson, Registration No. 47,652
Marc A. Boillot, Registration No. 56,164
AKERMAN SENTERFITT
Customer No. 40987
Post Office Box 3188
West Palm Beach, FL 33402-3188
Telephone: (561) 653-5000